Name: Thanawin Pattanaphol SEN-209: Database ID: 013243096

Question 1 – Database System Design for New Business

1) Propose a hybrid database architecture (SQL + NoSQL) for this system.

   A relatively simple database architecture for this project, can be mainly divided into three main parts. Relational Database, NoSQL Database, and Cache.

   a) Relational Database (SQL)

      The relational database will be mainly used for storing and managing structured, transactional data that needs consistency.

      In this scenario, it would be appropriate to use PostgreSQL or MySQL for managing SQL-based data.

   b) NoSQL Database

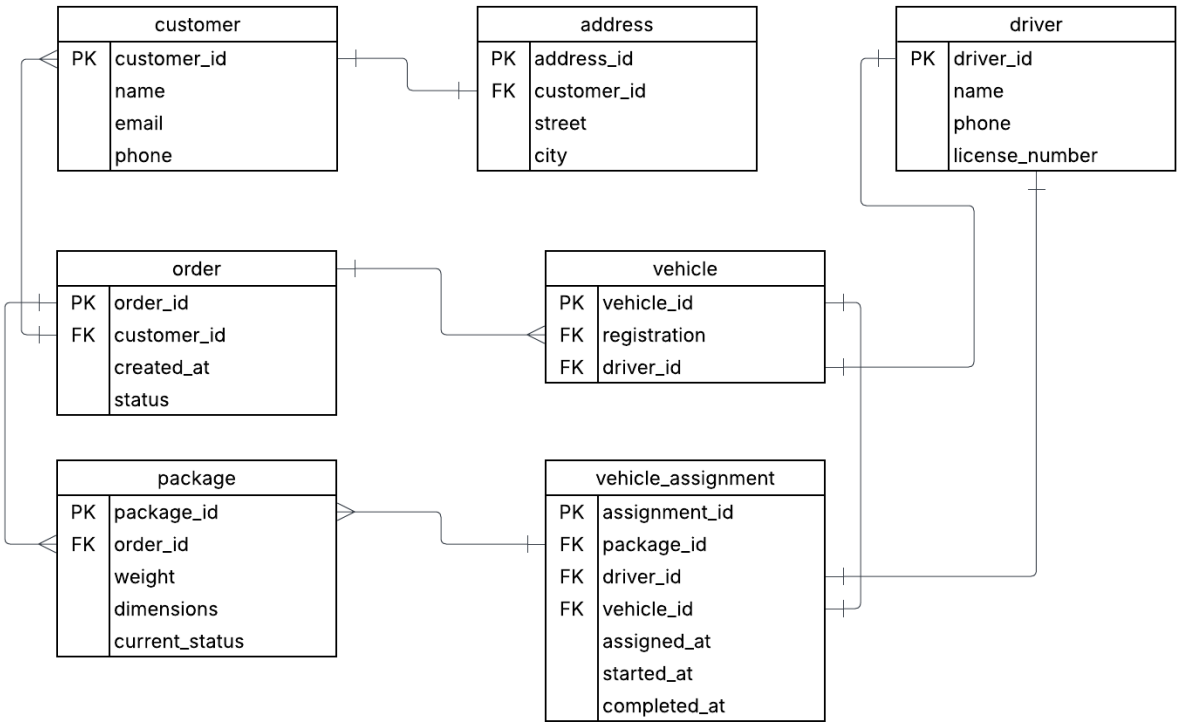
      NoSQL database, in this system, is mostly used for storing high-speed real-time data, such as, GPS locations, sensor updates, and analytics summaries.

      An optimal technology solution that can be used are tools such as, MongoDB, Cassandra or Elasticsearch.

   c) Cache

      Since this system requires frequent access of data or “live” data, it would be beneficial to use in-memory databases like Redis to keep frequently used data in memory for faster data read speeds.

2) Draw a simplified ER diagram for the relational portion.



3) Describe what data fits best in NoSQL and why.

In our system, the data that fits best with NoSQL are ones that have real-time updates, speedy reads / writes. They are ones, such as, location pings, device log streams, route data, driver data, search queries, geospatial indexes, and current-state / session caches.

All the mentioned data require fast writing and reading while also having no-fixed data schema or structure, which is essentially why NoSQL databases are appropriate for storing these types of data.

4) Suggest indexing strategies for performance optimization.

Important indexing strategies that will have a hugely positive impact on performance optimization can be categorized in to two main types, indexing strategies for the relational database and NoSQL / Time-series.

For relational databases (PostgreSQL, MySQL), having primary keys & foreign keys on every table; easier way to manage data relationships, create indexes for tables that are frequently used or queried.

For NoSQL databases, indexing strategies are ones, such as, composite primary keys, adjusting TTL, avoid heavy global secondary indexes and use design queries around partition keys, store geospatial prefixes as searchable field, and time-based compaction.

5) Identify one consistency challenge this system might face and you'd address it.

   A consistency challenge that this system can face are concurrent updates that may conflict with each other, e.g. package statuses, current locations (from multiple sources).

   Several strategies can be applied to mitigate said problems, e.g. defining canonical domains (canonical truth or transactional states), use single-write or coordinated writes for status changes (setting up primary writers for drivers to their assigned packages), versioning & idempotency (every update must have its version or sequence number and a timestamp), hybrid consistency rules (eventual consistency for non-critical views and strong consistency / ACID for financial-related, customer cancellations and final delivery confirmations in PostgreSQL), monitoring.

Question 2 - Concurrency, Transactions, and Security

1) Describe what could go wrong without proper transaction isolation.

   Thousands of students enroll at once and transactions are not properly isolated can lead to problems, such as
   a) Dirty reads, in which, one transaction read uncommitted changes from another.
   b) Lost update; two transactions update the same data, but one overwrites the other's changes.
   c) Non-repeatable Read; The same query returns different results within one transaction.
   d) Over-enrollment; Due to race conditions, the number of enrolled students exceeds the course capacity.

2) Compare two isolation level and explain their trade-offs in this context.

   The Read Committed Isolation Level, which is the default level in PostgreSQL, is when each query sees only the committed data, while other transactions updates become visible between statements. The benefits of this level is the fast speed and good concurrency of read and write speeds; the cons of this level is the risk of non-repeatable reads and over-enrollment if two transactions check seat availability at the same time.

   The Serializable Isolation Level, or the strictest isolation level in PostgreSQL, is when transactions have as if executed one at a time in sequences. The pros are how it prevents lost updates and over-enrollment, however, the trade-off that comes in is the slower speed due to transaction retries require larger computing power.

3) Suggest a locking or concurrency control strategy to prevent over-enrollment.

   A locking control strategy that can prevent over-enrollment is adding row-level locking, in which, locking the course row until the transaction ends, preventing others from reading outdated seat counts or updating simultaneously.

4) Describe one database security threat (e.g., injection or privilege misuse) and how you would mitigate it in this environment.

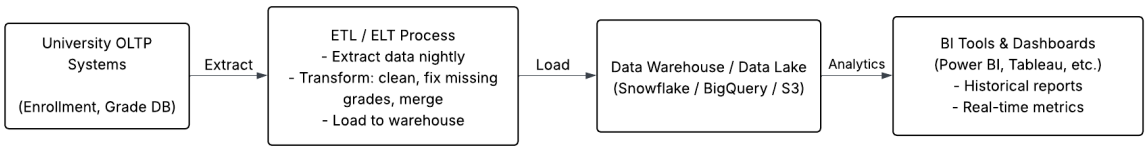
   A crucial security threat that may occur is SQL Injection. There are several ways to mitigate SQL injections, e.g. parameterized queries / prepared statements, validate all input, use least privilege roles, enable PostgreSQL role-based access control (RBAC), regular auditing & logging or all enrollment operations.

Question 3 – Analytical Design and Data Flow

1. Distinguish between OLTP and OLAP roles in this pipeline.

Aspects	**OLTP**	**OLAP**
Purpose	Handle daily operations like course registration, grade entry, payments.	Analyze large volumes of historical data for insights
Users	Students, staff, administrators.	Analysts, deans, planners, executives
Data Type	Current, detailed, real-time transactions.	Historical, summarized, multi-year data
Database Type	Relational DB (e.g. PostgreSQL, MySQL).	Data Warehouse (e.g. BigQuery, Snowflake, Redshift)
Operations	INSERT, UPDATE, DELETE	Complex SELECT, JOIN, GROUP BY, aggregations
Goal	Accuracy and speed of transactions	Deep analysis and trend discovery

2. Design a simplified data flow diagram showing how data moves from OLTP to Data Warehouse or Data Lake.



3. Propose two analytical queries (in SQL) that decision-makers could use.

   Query 1: Course Popularity Trend

```
SELECT
  c.course_name,
  COUNT(f.enrollment_id) AS total_enrollments
FROM Enrollment_Fact f
JOIN Course_Dim c ON f.course_id = c.course_id
GROUP BY c.course_name
ORDER BY total_enrollments DESC
LIMIT 5;
```

   Query 2: GPA Trend Over Time by Department

```
SELECT
  t.academic_year,
  d.department_name,
  ROUND(AVG(f.grade_points), 2) AS avg_gpa
FROM Enrollment_Fact f
JOIN Term_Dim t ON f.term_id = t.term_id
JOIN Course_Dim c ON f.course_id = c.course_id
JOIN Department_Dim d ON c.department_id = d.department_id
GROUP BY t.academic_year, d.department_name
ORDER BY t.academic_year, d.department_name;
```

4. Discuss one advantage and one limitation of using a cloud-based data warehouse for this project.

   The one advantage that cloud-based data warehouses has is scalability; warehouses can automatically scale storage and compute for large datasets and complex queries.

   The one limitation that cloud-based data warehouses has is its cost and latency management; continuous real-time queries and large-scale data scans can become expensive, therefore, the control of query is needed.

Question 4 – Backup, Recovery, and Disaster Readiness

1. Explain the exact recovery process to restore to the moment right before the crash (Point-in-Time Recovery).

   The recovery process would be the following.

   Stop the PostgreSQL service → Restore the last base backup (bring data back to its data at 11:00 PM yesterday) → Restore WAL or transaction logs → Create a recovery configuration (for PITR – set PostgreSQL into recovery mode) → Start PostgreSQL → Verify data consistency & remove recovery artifacts → Restart PostgreSQL (bring back to normal operation mode).

2. Define RTO and RPO for this case, and estimate reasonable values.

Term	Definition	Example for This Case	Estimated Value
RTO (Recovery Time Objective)	How long it takes to bring the database back online after failure.	Time to restore backup, replay WALs, verify system.	~30–60 minutes (depends on DB size & WAL volume)
RPO (Recovery Point Objective)	Maximum acceptable data loss (how old the data can be after recovery).	Since WAL archiving is continuous, the RPO approximately equal to time gap between last WAL archived and crash.	< 1–2 minutes (near-zero loss if WAL continuous)

3. Propose one improvement to the current backup strategy to minimize downtime in the future.

   Since our goal is to reduce the amount of downtime in this backup strategy, it is appropriate to setup a streaming replication or hot standby replica since it keeps a secondary standby server continuously updated from WAL stream which consequently reduces RTO from 30-60 minutes a few seconds and RPO to essentially zero since the standby is always online.

Question 5 – The Future of Databases

1. Compare traditional RDBMS, NoSQL, and vector databases in terms of data model, scalability, and typical use cases.

Feature	Trad. RDBMS	NoSQL Database	Vector Databases
Data Model	Structured databases Contains: tables, rows, columns (relational schema)	Flexible / unstructured Contains: key-value, document, graph, or column family	Numerical vector embeddings (multi-dimensional arrays) representing semantic meaning
Schema	Rigid (predefined schema)	Schema-less or dynamic	Schema-light (metadata + vector)
Scalability	Vertical scaling (add more CPU/RAM to one server)	Horizontal scaling (distributed nodes / sharding)	Horizontal and approximate search scaling (optimized for high-dimensional data)
Query Language	SQL	Can be JSON queries, APIs (e.g., MongoDB, Cassandra)	Vector similarity search (e.g., cosine similarity, Euclidean distance)
Transactions	ACID-compliant	BASE or eventual consistency	Usually not transactional as they

			focus on retrieval & similarity search
Typical Use Cases	Financial systems, ERP, student databases, e-commerce	Social media, IoT, big data, real-time analytics	AI applications or semantic search, recommendation systems, embeddings storage
Examples	PostgreSQL, MySQL, Oracle	MongoDB, Cassandra, DynamoDB	Pinecone, Milvus, PostgreSQL + pgvector, Weaviate

2. Discuss how AI-driven optimization (e.g., self-tuning queries or automatic indexing) could change the role of database administrators.

    AI-driven optimization provides self-tuning queries, automatic indexing, anomaly detection and resource prediction to traditional databases; this leads to the DBA shifting towards manual tuning and maintenance to data strategy, security, and governance oversight. They instead focus on data lifecycle, automation supervision, cloud cost optimization and integration with AI/analytics pipelines.

3. Propose one emerging trend (e.g., multi-model systems, data governance, or autonomous DBs) and explain its potential ethical or technical challenges.

    One emerging trend that is apparent in modern database systems is the increase in multi-model systems; their enhanced ability in handling multiple data types in one engine provides a much easier way of storing different types of data in one single database system, however, these pros still come with their cons / trade-offs.

    There can be several technical challenges with said system, e.g. complex query optimization (different data type combinations lead to advanced query planners, leading to difficult performance tuning), data consistency & versioning (maintaining ACID properties across distributed systems becomes even more complex than it is now), and resource management (more resource heavy, therefore, more usage of CPU / memory).

    Alongside technical challenges, come ethical challenges, e.g. data privacy in A.I. powered databases (vector databases often store data or sensitive data which can potentially end with leakage of private information), algorithmic transparency (tuning and optimization decisions become automated leading to organizations losing track of why certain actions were being done in the DBMS), and bias and accountability (AI-based systems often favor certain worklads or data sources, leading to biased query results or resource allocation).